Jada Thibodeaux

7-2 Project Two Submission

CS-320-T2644 Software Test Automation & QA

Professor Angelo Luo

December 13, 2021

The software development life cycle contains various stages, and one of the most important stages is testing. The whole point of testing is to ensure that the code meets all the requirements mandated and is free of errors. Testing is quite important before we edit and fix errors because what is the use of testing the code, if we will not edit it? Testing code is a vital step to see where we can improve at within improvement. Errors happen, so testing is the one step that is supposed to ensure we do not have errors. There are various forms of testing to help prevent the known and unknown problems in the software and tech world. Two are the most important types of testing are Static and Dynamic testing.

Static testing is testing that test software and work products with executing the actual code. The purpose of static testing is to find and remove errors and bugs in requirement documents before you translate them to code. This is all usually done in the development process before implementation. The whole point of this is to reduce the original source of error made into code. Essentially, static testing is a review within the requirements and rough draft portion of coding before the actual coding is done; a correct outline ensures a correct project, whereas an outline filled with error is project full of discrepancies.

Dynamic testing is the kind of testing that exercises the program. Dynamic testing can be described as testing when the code is executed. Software testing method tools such as Circle CI and Jenkins are used to test the dynamic of the code. Its purpose is to validate the input and output with the expected outcome of the software. After the dynamic testing is complete, the various components such as memory and CPU usage and performance are analyzed.

Static testing and dynamic testing are types of testing frequently used but are different in a few ways. Static testing is done by early review processing before the code is implemented. It is done at the beginning of the developing process to ensure there are no discrepancies within the requirements. Dynamic testing done at the end of the development process after the code is complete to find errors within the code. Dynamic testing tools used to check code optimizations are software like Circle CI and Jenkins.

Although both types of testing are different, it is important to utilize each method. Static testing is important in the beginning to make sure that there are no errors in the beginning. If there are errors in the requirements, once the code is complete, there will be evident errors in the dynamic testing portion. Both methods go hand in hand, causing one fault on another in a domino like fashion. For successful software, one method cannot be done without the other.

Graphical user interface, text, application, email

Description automatically generatedTesting is one of the most important ports in software development lifecycles. The purpose of testing is checking to see if the application’s code meets all the requirements and standards mandated by the client. Unit testing is testing that was created by developers to check the given Junit cases. It was a little tricky at first, and it did take me quite some time with constructing the code. When I began writing code for the beginning classes, I divided the complex parts by working on each class based on the tutorials in course’s modules. I did this so that it would help me cover all the requirements for each class. After some time, I am confident that my Junit code cases were sufficient and was able to see some of the case output from the application.

It took me a little longer to write the Junit cases because of all the various rules. I’m sure I had a few errors because I struggled with the organization and annotation of what goes where. For example, below, the test code that were part of a Junit, I have the correct code because of some annotation. Like Module Three’s coding annotations, I made sure to utilize the @Test annotation.

The Junit code was straightforward and concise for testing the application. With the Task.java class, is being tested by allotting new data then fetching the data given to it. If the test case can retrieve the same data, it will be on a pass or fail result. It’s just an extra test to see if the Task class is working correctly based on the given client requirements. I found this all to be the best and most efficient approach to the code.

In previous models, testing is an extremely important part of the Software Development Lifecycle. One style of testing that was done was Junit testing. In Milestone 3 I constructed a program that allowed contacts to be added, deleted, and modified. In this module I was able to implement the Junit5 testing. I also noticed the pattern between each milestone with the requirements. Each milestone had to construct certain apps that contained portions and those portions tests, usually containing about four classes. I think that there were a lot of assertTrue method cases between the fields and wanted values. This allowed me to validate that the fields were initialized correctly in each class.

I found that the assertTrue method is great to verify the field method given the rubric’s specifications, but there needs to be another structure implemented to tell whether or not the program can handle unknown output values. In order to do this, each object class has the needed checks within the constructors to alert the possible exceptions. Another testing method that can be implemented that I have not used is Test Pad, which allows developers to run multiple tests to make sure code works and meets the needed requirements. It can be used on multiple operating systems and the information can be used for other developers to look at, sort of as a log. I suppose this could be used with Junit, but there are also other tools that can be used to test various projects.